BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- BLURRY OR ILLEGIBLE TEXT
- SKEWED/SLATED IMAGES
- COLORED PHOTOS
- BLACK OR VERY DARK BLACK AND WHITE PHOTOS
- UNDECIPHERABLE GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 703 524 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
27.03.1996 Bulletin 1996/13

(51) Int. Cl.[6]: **G06F 3/00**, G06K 15/00

(21) Application number: **94202642.8**

(22) Date of filing: **13.09.1994**

(84) Designated Contracting States:
BE DE FR GB NL

(71) Applicant: **AGFA-GEVAERT**
naamloze vennootschap
B-2640 Mortsel (BE)

(72) Inventors:
- **Herregods, Marc**
  c/o Agfa-Gevaert N.V.
  B-2640 Mortsel (BE)
- **Tjantele, Dirk**
  c/o Agfa-Gevaert N.V.
  B-2640 Mortsel (BE)

(54) Variable data fields in a page description language

(57) A method of printing a set of documents which have a generally identical background image, and differ from each other by a small area, a page specific image region. A method is described to locate each page specific image region in a background image page layout. A master file stores this background image information along with positional parameters and a file reference to the page specific image region data. The page specific data are stored in one or more data files, generated by a database application program. The master file and page specific file are combined such that the background image is transformed to a bitmap only once, and the page specific data variously modify this bitmap to represent each individual page.

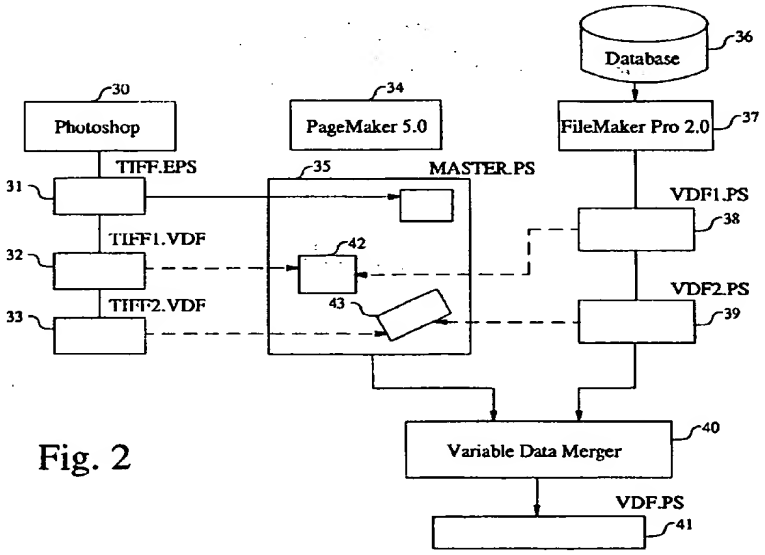

Fig. 2

EP 0 703 524 A1

Description

Field of the invention.

The present invention relates to a method for creating multiple documents having identical background image regions and document specific image regions containing individual data elements. The method can be used in desk top publishing systems and for professional printed matter e.g. for direct mailing or personalised copies.

Background of the invention.

For direct mailing purposes or in the production of personalised printed matter, it is necessary to print between ten or one thousand documents, having the same contents, except for a specific small area of the document. Usually, a document consists of one single sided sheet of paper, but such a document can also consist of several single sided or even double sided sheets. It is also possible that several pages of such a document must be printed or imposed on a single sheet of paper. One or more imposed sheets are then fold and/or assembled in a specific order to deliver a folder or booklet with the required layout. We will discuss here the problems that arise when individualised single sided sheets must be provided, although the current invention also solves these problems for the more complex configurations.

The most simple format of an individualised single sided sheet comprises a general text with open spaces. In the open spaces, the specific data are filled in per page. Traditionally, this is handled in the following way : the general text - indicated further on as the background image - is printed on a plurality of sheets, which are all identical. The batch printing can be done by an offset printer, by a photocopier or by a digital printer. The page specific information can be added immediately after the first printing pass or on a later moment. This can be per page by an individual tag stuck on the sheet, by hand writing, type-writing or by a printer coupled to a computer. Such a printer can be an impact printer, electrographic laser printer, inkjet printer etc. The problems for this method are the visible difference in writing and between the ink of the background image and the ink of the page specific data. Moreover, the page specific text is usually not properly aligned with the background text. The second pass to add the page specific data requires extra time and printing devices. If the background quality must be high, offset printing is required, which is very costly for small batches of individual copies. Another important drawback of this method is that only overwriting is possible. Nothing from the background image can be locally erased.

In the current digital output systems comprising a bitmap printer, for example in desktop applications, it is possible to generate a data stream for individual pages in a page description language. For each page to be printed, the data stream comprises a description of the background image and a description of the individual image. For each individual page, the data stream describing the background image and the specific data must be converted to a bitmap. If the background image is complex, this means an important burden for the raster image processor (RIP) generating the bitmap, while just a small portion on the sheet will be different from the previous sheet. Moreover, the transmission per sheet of the data stream describing the background data, can impose a large performance reduction on the total system. If the transmission goes over a network, this kind of print job imposes a tremendous load on the connection, thereby influencing the throughput of other tasks using the same network.

A method that alleviates the transmission problem is the creation of "forms". A page description language that supports the definition and use of forms is a Level 2 feature of the Postscript page description language. PostScript is a trade mark of Adobe Systems Inc. The PostScript language reference manual, 2nd edition ISBN 0-201-18127-4, chapter 4.7 on pages 172 to 175 describes the concept and the use of forms. A fixed template can be defined in a form, and the variable information can be painted on top of it. Each execution of the form will produce the same output. The graphical output of the form is saved in a cache. Each time the form is used, the saved output may be retrieved instead of re-executing the form's definition. The manual states that this can significantly improve performance when the form is used many times. The way of caching is implementation dependent. In most implementations, the cache stores an internal representation - a display list - that is converted to a bitmap each time the form is required. Anyhow, if a form contains an image, the whole image must be cached, which requires a substantial amount of memory. Moreover, the generation of a bitmap from the display list still requires as serious amount of work.

None of the above described methods give a satisfactory solution to the problems sketched herein before.

Objects of the invention.

It is therefore a first object of the invention to provide a method that generates high quality pages having an identical background image and a specific image on each page.

It is a further object of the invention that each page is generated in a single print pass.

It is a specific object of the invention that the computational effort to generate pages following the first page is substantially reduced with respect to the work required to generate the page specific image region.

Other objects will become apparent from the description hereinafter.

Summary of the invention

In accordance with the present invention, a method is disclosed for printing a plurality of pages, having an

identical background image region and at least one page specific image region, comprising the following steps :

a) generating a bitmap representation for said background image region and storing said bitmap in a bitmap memory means ;
b) saving in a cache memory means portions of said bitmap representation corresponding to each page specific image region;
c) generating a bitmap representation for at least one page specific image region and storing said page specific bitmap in said bitmap memory means ;
d) outputting the contents of said bitmap memory means to a marking engine for printing at least one page ;
e) restoring at least one said saved portion from said cache memory means to said bitmap memory means ;
f) repeating steps c) to e) until said plurality of pages is printed.

The feature that the background region is directly converted to a bitmap representation of the background image, and not stored in some internal format in a separate cache memory for later retrieval as in the implementation of the PostScript "form" command has several advantages :

- if the internal format is not a bitmap format, the execution of the "execform" command requires the generation of the bitmap ;
- if the internal format is a bitmap format, a large amount of cache memory is required for storing the form, apart from the memory required for the bitmap ; moreover the "form"-functionality requires that an extra non-bitmap copy is retained to allow scaling, rotation and other graphical state settings ;
- even if the form is stored in bitmap format, the contents of the separate cache memory must be transmitted to the bitmap memory each time the "execform" command is executed. In a 400 dpi A3-size colour printer, this can require a data copying of 120 MByte, which requires 3 seconds in a system having a bus bandwidth of 40 MByte per second.

According to the method of the current invention, the bitmap portions - corresponding to the page specific image regions - of the bitmap memory means that must be saved, are usually substantially smaller than the whole bitmap. This has the advantage that just a small amount of memory is required to save the bitmap portions. Moreover, the saving operation and later on the restoring operations do not put a big data transmission burden on the raster image processor.

The steps c), d) and e) that must be executed for each individual page are mainly restricted to the bitmap generation for the page specific data. The bitmap data for the page specific image regions can directly be written into the bitmap memory, as if one single page must be printed. As such, no specific processing for this bitmap region or extra processing time is necessary, enabling the system to make use of the high performance processing of usual bitmap generation. The outputting of the bitmap to the marking engine - step d) - must be done in any case. A third essential step e) is the restoration of the bitmap data representing the background image. This must be accomplished at least for the bitmap portion that will be different on the next page. The advantage of this restoration is that e.g. text can always be overlaid on the true background, such that this remains visible where no text is painted. On the other hand, a box containing the text can also be put over the background, whatever is required by the layout. Even the size of this box could vary between consecutive sheets.
Such bitmap portions are usually substantially smaller than the background bitmap, such that the transmission requires a small amount of work, which can be done rapidly. Moreover, the restoration of the background bitmap is accomplished by a simple copy operation, involving no other computational effort. A copy operation can be dramatically optimized, based on hardware capabilities or sometimes by specific software implementations in machine language or microcode. To optimise this transmission, preferentially each bitmap portion consists of a rectangular area with horizontal and vertical sides, which fully overlaps one or more page specific image regions. A page specific image region can be a rectangle rotated over e.g. 30 degrees. In that case, the bitmap portion is preferentially the smallest horizontally oriented rectangle, enclosing the rotated rectangle. The page specific image region may also be delineated by a polygon or any other closed path. The bitmap portion is also in that case preferentially the smallest enveloping horizontal rectangle.

The sheets that must be printed are usually sheets of paper. The sheets can be printed single sided or double sided. By a plurality is meant that at least two sheets are printed, having the same background image. Usually a sequence of say hundred of sheets is printed, having the same background image. It is also possible that each individualised sheet must be printed twice or more times, when two or more identical copies are required for each specific page. It is also possible that the same "background image" appears e.g. four times on one sheet, and that on that sheet four individual pages are prepared. The four "background images" form in that case one large "background image" in the teaching of this invention and the sheet then contains four, eight, .. sheet specific image regions. A background data stream describing such type of pages is usually obtained by "step and repeat" imposition, in the case that each required sheet fits two or more times on one printed sheet.

It is also possible that each individual document consists of two or more sheets. In that case, preferentially the first pages of each individual document are generated and output to the printer first, followed by the second

The raster image processor (RIP) 23 can be an Agfa CR-A system, CR-A is a trade name of Agfa-Gevaert N.V. in Mortsel Belgium. The CR-A RIP supports PostScript level 1 and an extra feature to enable the method of the current invention. The system comprises two Motorola MC68040 processors, operating at a clock rate of 33 MHz, it further contains an Agfa specific CVI interface board to drive the marking engine 23. The system further comprises a SCSI interface board towards a 340 MByte SCSI compatible hard disk 26, and an ethernet connection 27 to hook the raster image processor 23 up to the same network as to which the workstation 21 is connected to. The CR-A RIP further comprises a random access memory board (not shown), capable to store 128 MByte, with an access time of 80 nanoseconds.

The marking engine 24 is preferentially an XC305 or XC315 Agfa Colour Copier, both trade names of Agfa-Gevaert N.V. in Mortsel, Belgium.

^A way for carrying out the invention is now described in conjunction with Fig. 2. If the background image comprises digital images, obtained for example from contone colour images on photo prints, the digital image source files can for instance be generated by the above mentioned Photoshop application program 30, and saved according to the TIFF (Tag Image File Format, trade mark of Aldus Inc.) specification on hard disk in a file called TIFF.EPS 31. For each page specific image region preferentially a separate dummy TIFF file, with the extension .VDF is created by Photoshop. The dummy file 30 for the first page specific image region 42 can be called TIFF1.VDF, for the second page specific image region 43 TIFF2.VDF 33 etc. The contents of these TIFFx.VDF image files don't really matter, because their contents will not appear in the final VDF.PS combined data stream 42. Preferentially they represent low resolution images, such that they do not add to much data to the MASTER.PS file. Alternatively, images comprising templates for the real page specific regions can be created, to allow proper alignment with the background image, as will be discussed below.

In a next step, a page layout program 34 such as QuarkXPress or PageMaker 5.0 is invoked. Usually, first a text to be printed on the sheet is imported and shown on the screen of the workstation according to the selected options such as character font, size and other characteristics. All background composition images are then "imported" via the page layout program. This means that a low resolution version of the background composition image as required is shown on the video monitor of the workstation 21, but the whole image is not actually accessed. Each background composition image can be positioned relative to the text within the page layout, mainly with respect to translation, rotation and scaling. If required, more text can be added to annotate the images, text can be reorganised etc. Also the TIFFx.VDF (32, 33) images, representing the page specific image regions, must be properly positioned within the page layout. If each TIFFx.VDF image is just a dummy image, the rectangular area of that image can be properly positioned with respect to the rest of the layout by translation, resizing each side of the rectangle and rotation, over any angle. Preferentially, each TIFFx.VDF file contains an image representation of one page specific image region. If the first page specific image region would contain a person's name, then TIFF1.VDF preferentially represents an image of an arbitrary name, in the font and size as required in the final printed sheet. As such, the rectangular box enclosing that name and occupied by the TIFF1.VDF image can be properly positioned and aligned with the surrounding text in the background image. This rectangular box must not be resized in any dimension, because this would obscure the relation between the template on the video monitor of the workstation and the final result. Another way to properly along the page specific image region within the background image is to incorporate part or the whole test line, where the page specific data are located, in the region. The region is positioned as correctly as possible in a first pass, and after printing one document, and measuring the dislocation, re-positioning the region.

The page layout program will generate an output data stream 35 in a selected page description language. Preferentially, the PostScript Level 1 output format is selected. The output data stream is preferentially directed to a file on hard disk, which we will call the master file MASTER.PS. This master file gives a full description of the background image. The image data, both representing the background composition images and the dummy templates for the page specific image data, may be effectively included in the master file, preferentially in a low resolution format, but are also referenced by it through the file name. The way how an image is referenced to in a PostScript file, is defined by the OPI (Open Prepress Interface, trade mark of Aldus Inc.) standard, as described in the Open Prepress Interface Specification 1.2. In Fig. 3, we show an excerpt from a PostScript output of PageMaker 5.0. The comments - having a leading % character - define according to the OPI standard the image filename as TIFF1.VDF in the "%ALDImageFileName: "-field and further indicate that the image data are in TIFF format. The dimensions of the image are indicated as 142 pixels by 142 lines, and the rectangular area that the image will occupy in the layout is given as measured in points (1"/72) under the comment "%ALDImagePosition: ". This means that the size of the image and its location within the layout is given. The image data is inserted just after the "%%BeginData" comment line. The "is CL^" command will consume these data such that they are directed to the correct positions within the bitmap. The rectangular area delineated by the four corner points given in the "%ALDImagePosition" comment defines the positional parameters for the page specific image region, such as location, shape, size and orientation of the region. All page specific data must fit within this area. As will be described below, everything page specific outside this area will be clipped.

Once the layout for the background image, along with the positional parameters for the page specific

image regions, is fixed, the information contents for each individual page specific image region on each sheet must be generated. As shown in Fig. 2, this is preferentially done by a database application program 37 such as FileMaker Pro version 2.0. In this program, a set of records can be imported by manual entry or by accessing data on a database file 36. Each record contains individual fields. A record contains for example the information about a client. A first field in the record gives his last name, the next field his first name, the next field his address etc. The application program 37 allows to pick some fields and format the data in a specific layout. According to the above example, one could define a layout containing two lines : the first line represents the contents of the last name field followed by the first name field and the second line represents the contents of the address field. The application program 37 can now generate a first ASCII PostScript Level 1 compatible page specific data file 38 - which we will call VDF1.PS - that contains the thus formatted representation of all or a selection of some records from the database, without cover page inclusion. If that file VDF1.PS would be sent to a PostScript printer, one page for each record would be generated, each page containing two lines, formatted as described above. The text lines, graphical data or image in VDF1.PS must be positioned relative to a known reference, e.g. the lower left corner of each page, which is the origin position for the PostScript interpreter. This lower left corner will then be positioned upon the lower left corner of the rectangular page specific image region within the background image.

The VDF1.PS could reflect the address section of a letter, while further down the letter just the last name must be inserted in the background image. For that purpose, a second page specific PostScript data file VDF2.PS 39 can be generated, extracting from the database the same records as VDF1.PS, but formatting just one line containing each time the last name field of the record. Printing VDF2.PS would generate the same amount of pages as with printing VDF1.PS, but on each page just the last name would appear.

For every dummy file TIFFx.VDF (32, 33), a corresponding page specific data file VDFx.PS (38, 39) must be generated. Usually the number of TIFFx.VDF files is the same as the number of VDFx.PS files. It is possible however that the contents of one VDFx.PS file can be used for two different page specific image regions. If for these regions a different dummy file TIFFx.VDF and TIFFy.VDF were generated, it may be appropriate to generate just one VDFx.PS. In this case it could have been also appropriate to reference in the MASTER.PS PostScript file two times to the TIFFx.VDF file.

Usually all page specific image files VDFx.PS, VDFy.PS, ... represent the same amount of pages. As such, the first sheet of the final printed output would contain data corresponding to the first page section described in VDFx.PS and the first page section described in VDFy.PS. The same applies for the second and all subsequent pages. It is possible however to generate a VDFx.PS file describing N+N pages, and a VDFy.PS file describing just two pages. The first printed sheet would then contain the contents of VDFx.PS page section 1 and VDFy.PS page section 1 ; the second page would contain the contents of VDFx.PS page section 2 and VDFy.PS page section 2 ; the third page would contain the contents of VDFx.PS page section 3 and VDFy.PS page section 1 again, because VDFy.PS was exhausted and is cyclically repeated ; the fourth page would contain the contents of VDFx.PS page section 4 and VDFy.PS page section 2, etc.

Once both the master file MASTER.PS 35 and the page specific image files VDFx.PS 38, 39 are created, these files can be combined to yield the desired result. Therefore a variable data field application program or variable data merger 40 is invoked, offering the following options.

The number of copies for each individual sheet can be selected. If every document needs two or more identical copies, this is set by this number of copies. The paper size on which the background layout must be printed can also freely be selected. All images referenced to in the OPI compatible commands must be retrieved and explicitly included. These dummy images, referenced by a file extension .VDF must not be included as such but must be handled as described below. The variable data merger will generate an ASCII PostScript Level 1 compatible combined data stream 41. This data stream can be directed to a file, e.g. on hard disk or directly to a printer system, that accepts a PostScript page description language and prints the images represented by the language. If the data stream is directed to a file, this file can be stored temporarily on a storage medium and be sent at a later date be sent to a PostScript compatible printing system.

The variable data merger 40 will analyze the MASTER.PS file 35 and locate all the occurrences of a reference to a file with extension .VDF. According to the above example, the program will find TIFF1.VDF, TIFF2.VDF. The program will prompt the operator for the substitution of the TIFF1.VDF reference. According to the above example, the operator would introduce the filename VDF1.PS. The program then prompts for the substitution of TIFF2.VDF ; the operator will introduce VDF2.PS, etc.

According to the PostScript code in Fig. 4, we will describe now how the variable data merger 40 processes the different input files : one MASTER.PS file 35, several VDFx.PS files 38, 39 and possibly different TIFF-files 31 representing images that must be included in the background image. Therefore we need to recall how a PostScript file is structured according to the DSC recommendations.

According to DSC, as defined in the Document Structuring Conventions in the above mentioned PostScript language reference manual on pages 611 to 708, a PostScript file contains three main sections :

- a prolog script ;
- a page section ; and

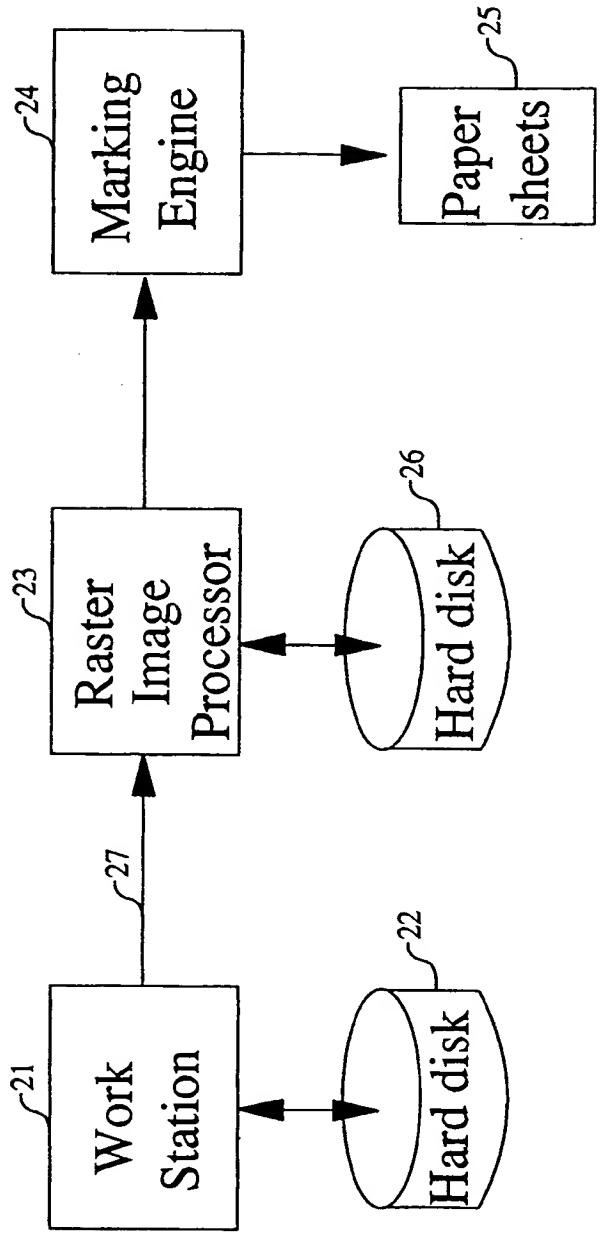

Fig. 1

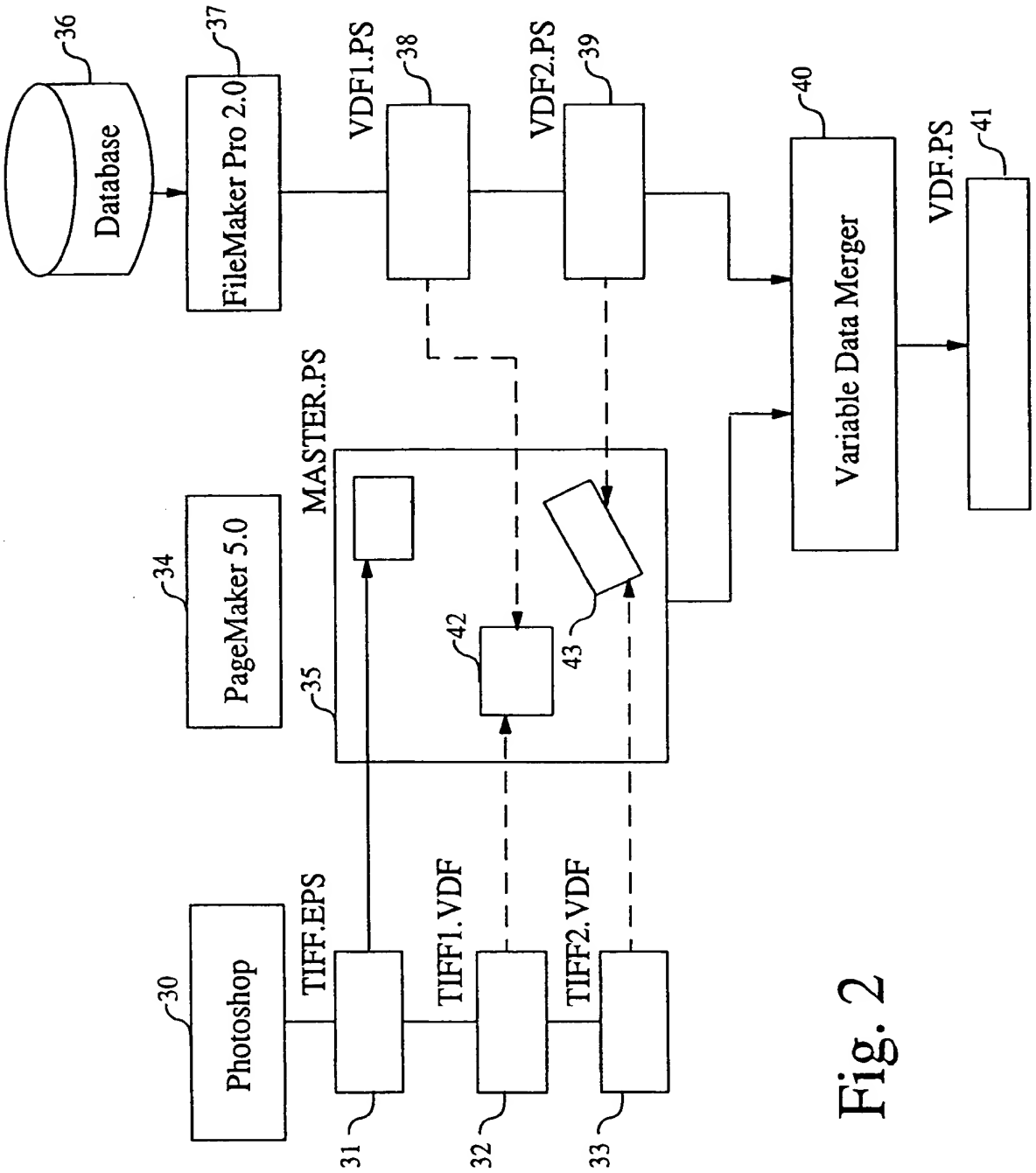

Fig. 2

```
IMdict begin
/AD_ImMatrix matrix def AD_ImMatrix currentmatrix pop
tvs^
%ALDImageFileName: Macintosh HD:Data:TIFF Files:TIFF1.VDF
%ALDImageDimensions: 142 142
%ALDImageCropRect: 0 0 142 142
%ALDImageCropFixed: 0.00000 0.00000 142.00000 142.00000
%ALDImagePosition: 396.0 650.5 396.0 792.5 538.0 792.5 538.0 650.5
%ALDImageResolution: 72.000 72.000
%ALDImageColorType:  Process
%ALDImageColor: 0.000 0.000 0.000 1.000 (Black)
%ALDImageTint:  1.000
%ALDImageOverprint:  false
%ALDImageType: 1 8
%ALDImageGrayMap: 0 262 524 786 1049 etc...
%%BeginObject: image
AD_ImMatrix setmatrix
[20.000000 0.000000 0.000000 20.000000 7920.000000 -3600.000000] concat
true 142 142 8 false 0.000 0.000 0.000 1.000 (Black) 1.000 true
%%BeginData: 20171 Binary Bytes
isCL^
%%EndData
%%EndObject
end
```

Fig. 3

```
userdict /__AGFA__Copies  1 put

/__AGFA_VDF_DICT 100 dict def

 __AGFA_VDF_DICT begin
  /__AGFA_VDF_CTM matrix defaultmatrix def
  /PageSizeRequest {
    initgraphics
    __AGFA_VDF_DICT begin
      __AGFA_VDF_DICT /__AGFA_VDF_BOUND known
        {__AGFA_VDF_BOUND aload pop newpath moveto lineto lineto lineto
         closepath clip} if
        __AGFA_VDF_CTM setmatrix  end
   } def
 end

/__AGFA__RedefIfKnown {
        currentdict begin
          [0 {== __AGFA_VDF_DICT begin PageSizeRequest end} /exec load]
          2 copy exch 50 string cvs  0 exch put cvx def
        end
 } def

[/a3 /a4 /A4 /a4small /a5 /A5 /b5 /B5
 /Letter /letter /lettersmall /note /legal /11x17 /ledger]
 {__AGFA__RedefIfKnown} forall

 statusdict begin
        [ /lettertray /11x17tray /ledgertray /legaltray
          /statementtray /executivetray /a3tray /a4tray  /b4tray /b5tray]
        {__AGFA__RedefIfKnown} forall

        /setpapertray  dup currentdict exch known
          {{pop __AGFA_VDF_DICT begin PageSizeRequest end} bind def}
          {userdict exch {pop __AGFA_VDF_DICT begin PageSizeRequest end}
           bind put} ifelse
 end

/setpagedevice {pop __AGFA_VDF_DICT begin PageSizeRequest end} def

__AGFA_VDF_DICT begin

  /__AGFA_SAVE_CONTEXT {
    __AGFA_VDF_DICT begin dup 3 dict def load end
    /_numdicts countdictstack put
    __AGFA_VDF_DICT begin       /__AGFA_ContextSave save def end
  } def

  /__AGFA_RESTORE_CONTEXT {
    __AGFA_VDF_DICT begin load
      begin
        /_numdicts load
      end
    end
    countdictstack exch sub dup 0 gt
      {{end} repeat} if
    __AGFA_VDF_DICT begin __AGFA_ContextSave restore end
  } def
end
```

Fig. 4.A

```
/_AGFA_SHOWPAGE {userdict begin
        /#copies __AGFA__Copies def
        systemdict
        /showpage get exec
        __AGFA_VDF_DICT begin
        /__AGFA_VDF_CTM matrix defaultmatrix def
        end}
def

/_AGFA_COPYPAGE {userdict begin
        /#copies __AGFA__Copies def
        systemdict
        /copypage get exec
        end}
def

/_AGFA_CLEAR_RECT {
  statusdict begin
    statusdict /setvariabledatabox known
        {8 {pop} repeat}
        {systemdict begin matrix defaultmatrix setmatrix end initclip
          newpath moveto lineto lineto lineto closepath
          currentgray 1 setgray fill setgray}
      ifelse
}  def

/_AGFA_MAKE_TRANS_FROM_RECT {
  systemdict begin matrix defaultmatrix setmatrix end
  8 copy 8 array astore
  __AGFA_VDF_DICT begin /__AGFA_VDF_BOUND exch def end
  8 copy newpath moveto lineto lineto lineto closepath clip
  4{pop}repeat 4 copy
  exch 4 1 roll exch sub 3 1 roll exch sub exch atan 5 1 roll
  pop pop translate rotate
  __AGFA_VDF_DICT begin /__AGFA_VDF_CTM matrix currentmatrix def end
} def

/_AGFA__SAVE_VDF_BOX
  {statusdict begin statusdict /setvariabledatabox known
    {matrix defaultmatrix setmatrix setvariabledatabox }
    {pop pop pop pop pop (ppppp) ==}
  ifelse end
} def

/_AGFA_SHOWPAGE_TO_NOTHING {/showpage {} def } def

_AGFA_SHOWPAGE_TO_NOTHING

% PageMaker Prolog section - PageMaker Page section 1

__AGFA_VDF_DICT begin /savecon /__AGFA_SAVE_CONTEXT load end exec

/__AGFA_VDF_RECT_11 {
  396.000 650.500 396.000 792.500 538.000 792.500 538.000 650.500
} def

1 396 650 538 793 _AGFA__SAVE_VDF_BOX
```

Fig. 4.B

```
% Start loop over different database records
__AGFA_VDF_RECT_11 _AGFA_CLEAR_RECT
gsave
__AGFA_VDF_RECT_11 _AGFA_MAKE_TRANS_FROM_RECT
% FileMaker VDFx.PS Prolog section ; Page section 1 ; Trailer section
grestore
_AGFA_COPYPAGE
% Second iteration
__AGFA_VDF_RECT_11 _AGFA_CLEAR_RECT
gsave
__AGFA_VDF_RECT_11 _AGFA_MAKE_TRANS_FROM_RECT
% FileMaker VDFx.PS Prolog section ; Page section 2 ; Trailer section
grestore
_AGFA_COPYPAGE
% Third iteration
__AGFA_VDF_RECT_11 _AGFA_CLEAR_RECT
gsave
__AGFA_VDF_RECT_11 _AGFA_MAKE_TRANS_FROM_RECT
% FileMaker VDFx.PS Prolog section ; Page section 3 ; Trailer section
grestore
_AGFA_COPYPAGE
% Last iteration
__AGFA_VDF_RECT_11 _AGFA_CLEAR_RECT
gsave
__AGFA_VDF_RECT_11 _AGFA_MAKE_TRANS_FROM_RECT
% FileMaker VDFx.PS Prolog section ; Last Page section ; Trailer section
grestore
_AGFA_SHOWPAGE
__AGFA_VDF_DICT begin /savecon /__AGFA_RESTORE_CONTEXT load end exec
% Trailer section of PageMaker output MASTER.PS
```

Fig. 4.C

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 20 2642

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
A	GB-A-2 220 511 (CANON K.K.) * claims *	1,7	G06F3/00 G06K15/00
A	US-A-5 104 245 (OGURI ET AL.) * column 1, line 67 - column 3, line 8; figures 2-4 *	1,7	
A	PATENT ABSTRACTS OF JAPAN vol. 16, no. 250 (M-1262) 8 June 1992 & JP-A-04 059 372 (FUJITSU LTD.) 26 February 1992 * abstract *		
A	EP-A-0 131 966 (KANZAKI PAPER MANUFACTURING CO.)		
A	PATENT ABSTRACTS OF JAPAN vol. 18, no. 41 (M-1546) 21 January 1994 & JP-A-05 270 093 (CANON INC.) 19 October 1993 * abstract *		
			TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06K

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	3 March 1995	Gelebart, Y

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)